1. **Write a program to read and display n numbers using an array.**

```
#include <stdio.h>
#include <conio.h>
int main()
{
int i, n, arr[20];
printf("\n Enter the number of elements in the array : ");
scanf("%d", &n);
for(i=0;i<n;i++)
{
        printf("\n arr[%d] = ", i);
        scanf("%d",&arr[i]);
}
printf("\n The array elements are ");
for(i=0;i<n;i++)
printf("\t %d", arr[i]);
return 0;
}
```

2. **Write a program to find the mean of n numbers using arrays.**

```
#include <stdio.h>
#include <conio.h>
int main()
{
int i, n, arr[20], sum =0;
float mean = 0.0;
printf("\n Enter the number of elements in the array : ");
scanf("%d", &n);
for(i=0;i<n;i++)
{
printf("\n arr[%d] = ", i);
scanf("%d",&arr[i]);
}
for(i=0;i<n;i++)
sum += arr[i];
mean = (float)sum/n;
printf("\n The sum of the array elements = %d", sum);
```

```
printf("\n The mean of the array elements = %.2f", mean);
return 0;
}
```

3. **Write a program to print the position of the smallest number of n numbers using arrays.**

```
#include <stdio.h>
#include <conio.h>
int main()
{
int i, n, arr[20], small, pos;
printf("\n Enter the number of elements in the array : ");
scanf("%d", &n);
printf("\n Enter the elements : ");
for(i=0;i<n;i++)
scanf("%d",&arr[i]);
small = arr[0]
pos =0;
for(i=1;i<n;i++)
{
if(arr[i]<small)
{
small = arr[i];
pos = i;
}
}
printf("\n The smallest element is : %d", small);
printf("\n The position of the smallest element in the array is : %d", pos);
return 0;
}
```

4. **Write a program to find the second largest of n numbers using an array.**

```
#include <stdio.h>
#include <conio.h>
int main()
{
int i, n, arr[20], large, second_large;
```

```
printf("\n Enter the number of elements in the array : ");
scanf("%d", &n);
printf("\n Enter the elements");
for(i=0;i<n;i++)
scanf("%d",&arr[i]);
large = arr[0];
for(i=1;i<n;i++)
{
if(arr[i]>large)
large = arr[i];
}
second_large = arr[1];
for(i=0;i<n;i++)
{
if(arr[i] != large)
{
if(arr[i]>second_large)
second_large = arr[i];
}
}
printf("\n The numbers you entered are : ");
for(i=0;i<n;i++)
printf("\t %d", arr[i]);
printf("\n The largest of these numbers is : %d",large);
printf("\n The second largest of these numbers is : %d",second_large);
return 0;
}
```

5. **Write a program to perform Push, Pop, and Peek operations on a stack.**

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define MAX 3 // Altering this value changes size of stack created
int st[MAX], top=-1;
void push(int st[], int val);
int pop(int st[]);
int peek(int st[]);
```

```
void display(int st[]);
int main(int argc, char *argv[]) {
int val, option;
do
{
printf("\n *****MAIN MENU*****");
printf("\n 1. PUSH");
printf("\n 2. POP");
printf("\n 3. PEEK");
printf("\n 4. DISPLAY");
printf("\n 5. EXIT");
printf("\n Enter your option: ");
scanf("%d", &option);
switch(option)
{
case 1:
printf("\n Enter the number to be pushed on stack: ");
scanf("%d", &val);
push(st, val);
break;
case 2:
val = pop(st);
if(val != -1)
printf("\n The value deleted from stack is: %d", val);
break;
case 3:
val = peek(st);
if(val != -1)
printf("\n The value stored at top of stack is: %d", val);
break;
case 4:
display(st);
break;
}
}while(option != 5);
return 0;
}
```

```
void push(int st[], int val)
{
if(top == MAX-1)
{
printf("\n STACK OVERFLOW");
}
else
{
top++;
st[top] = val;
}
}
int pop(int st[])
{
int val;
if(top == -1)
{
printf("\n STACK UNDERFLOW");
return -1;
}
else
{
val = st[top];
top--;
return val;
}
}
void display(int st[])
{
int i;
if(top == -1)
printf("\n STACK IS EMPTY");
else
{
for(i=top;i>=0;i--)
printf("\n %d",st[i]);
printf("\n"); // Added for formatting purposes
```

```
}
}
int peek(int st[])
{
if(top == -1)
{
printf("\n STACK IS EMPTY");
return -1;
}
else
return (st[top]);
}
```

6. Write a program to calculate the factorial of a given number.

```
#include <stdio.h>
int Fact(int); // FUNCTION DECLARATION
int main()
{
int num, val;
printf("\n Enter the number: ");
scanf("%d", &num);
val = Fact(num);
printf("\n Factorial of %d = %d", num, val);
return 0;
}
int Fact(int n)
{
if(n==1)
return 1;
else
return (n * Fact(n–1));
}
```

7. **Write a program to calculate the GCD of two numbers using recursive functions**.

```
#include <stdio.h>
int GCD(int, int);
int main()
```

```
{
int num1, num2, res;
printf("\n Enter the two numbers: ");
scanf("%d %d", &num1, &num2);
res = GCD(num1, num2);
printf("\n GCD of %d and %d = %d", num1, num2, res);
return 0;
}
int GCD(int x, int y)
{
int rem;
rem = x%y;
if(rem==0)
return y;
else
return (GCD(y, rem));
}
```

8. **Write a program to calculate exp(x,y) using recursive functions.**

```
#include <stdio.h>
int exp_rec(int, int);
int main()
{
int num1, num2, res;
printf("\n Enter the two numbers: ");
scanf("%d %d", &num1, &num2);
res = exp_rec(num1, num2);
printf ("\n RESULT = %d", res);
return 0;
}
int exp_rec(int x, int y)
{
if(y==0)
return 1;
else
return (x * exp_rec(x, y–1));
}
```

9. **Write a program to print the Fibonacci series using recursion.**

```
#include <stdio.h>
int Fibonacci(int);
int main()
{
int n, i = 0, res;
printf("Enter the number of terms\n");
scanf("%d",&n);
printf("Fibonacci series\n");
for(i = 0; i < n; i++ )
{
res = Fibonacci(i);
printf("%d\t",res);
}
return 0;
}
int Fibonacci(int n)
{
if ( n == 0 )
return 0;
else if ( n == 1 )
return 1;
else
return ( Fibonacci(n–1) + Fibonacci(n–2) );
}
```

10. Write a program to implement a linear queue.

```
#include <stdio.h>
#include <conio.h>
#define MAX 10 // Changing this value will change length of array
int queue[MAX];
int front = -1, rear = -1;
void insert(void);
int delete_element(void);
int peek(void);
void display(void);
int main()
```

```
{
int option, val;
do
{
printf("\n\n ***** MAIN MENU *****");
printf("\n 1. Insert an element");
printf("\n 2. Delete an element");
printf("\n 3. Peek");
printf("\n 4. Display the queue");
printf("\n 5. EXIT");
printf("\n Enter your option : ");
scanf("%d", &option);
switch(option)
{
case 1:
insert();
break;
case 2:
val = delete_element();
if (val != -1)
printf("\n The number deleted is : %d", val);
break;
case 3:
val = peek();
if (val != -1)
printf("\n The first value in queue is : %d", val);
break;
case 4:
display();
break;
}
}while(option != 5);
getch();
return 0;
}
void insert()
{
```

```
int num;
printf("\n Enter the number to be inserted in the queue : ");
scanf("%d", &num);
if(rear == MAX-1)
printf("\n OVERFLOW");
else if(front == -1 && rear == -1)
front = rear = 0;
else
rear++;
queue[rear] = num;
}
int delete_element()
{
int val;
if(front == -1 || front>rear)
{
printf("\n UNDERFLOW");
return -1;
}
else
{
val = queue[front];
front++;
if(front > rear)
front = rear = -1;
return val;
}
}
int peek()
{
if(front==-1 || front>rear)
{
printf("\n QUEUE IS EMPTY");
return -1;
}
else
{
```

```
return queue[front];
}
}
void display()
{
int i;
printf("\n");
if(front == -1 || front > rear)
printf("\n QUEUE IS EMPTY");
else
{
for(i = front;i <= rear;i++)
printf("\t %d", queue[i]);
}
}
```

11. Write a program to search an element in an array using the linear search technique.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define size 20 // Added so the size of the array can be altered more easily
int main(int argc, char *argv[]) {
int arr[size], num, i, n, found = 0, pos = -1;
printf("\n Enter the number of elements in the array : ");
scanf("%d", &n);
printf("\n Enter the elements: ");
for(i=0;i<n;i++)
{
scanf("%d", &arr[i]);
}
printf("\n Enter the number that has to be searched : ");
scanf("%d", &num);
for(i=0;i<n;i++)
{
if(arr[i] == num)
{
```

```
found =1;
pos=i;
printf("\n %d is found in the array at position= %d", num,i+1);
/* +1 added in line 23 so that it would display the number in
the first place in the array as in position 1 instead of 0 */
break;
}
}
if (found == 0)
printf("\n %d does not exist in the array", num);
return 0;
}
```

12. Write a program to search an element in an array using binary search.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define size 10 // Added to make changing size of array easier
int smallest(int arr[], int k, int n); // Added to sort array
void selection_sort(int arr[], int n); // Added to sort array
int main(int argc, char *argv[]) {
int arr[size], num, i, n, beg, end, mid, found=0;
printf("\n Enter the number of elements in the array: ");
scanf("%d", &n);
printf("\n Enter the elements: ");
for(i=0;i<n;i++)
{
scanf("%d", &arr[i]);
}
selection_sort(arr, n); // Added to sort the array
printf("\n The sorted array is: \n");
for(i=0;i<n;i++)
printf(" %d\t", arr[i]);
printf("\n\n Enter the number that has to be searched: ");
scanf("%d", &num);
beg = 0, end = n-1;
while(beg<=end)
```

```
{
mid = (beg + end)/2;
if (arr[mid] == num)
{
printf("\n %d is present in the array at position %d", num, mid+1);
found =1;
break;
}
else if (arr[mid]>num)
end = mid-1;
else
beg = mid+1;
}
if (beg > end && found == 0)
printf("\n %d does not exist in the array", num);
return 0;
}
int smallest(int arr[], int k, int n)
{
int pos = k, small=arr[k], i;
for(i=k+1;i<n;i++)
{
if(arr[i]< small)
{
small = arr[i];
pos = i;
}
}
return pos;
}
void selection_sort(int arr[],int n)
{
int k, pos, temp;
for(k=0;k<n;k++)
{
pos = smallest(arr, k, n);
temp = arr[k];
```

```
arr[k] = arr[pos];
arr[pos] = temp;
}
}
```

13. Write a program to sort an array using insertion sort algorithm.

```
#include <stdio.h>
#include <conio.h>
#define size 5
void insertion_sort(int arr[], int n);
int main()
{
int arr[size], i, n;
printf("\n Enter the number of elements in the array: ");
scanf("%d", &n);
printf("\n Enter the elements of the array: ");
for(i=0;i<n;i++)
{
scanf("%d", &arr[i]);
}
insertion_sort(arr, n);
printf("\n The sorted array is: \n");
for(i=0;i<n;i++)
printf(" %d\t", arr[i]);
return 0;
}
void insertion_sort(int arr[], int n)
{
int i, j, temp;
for(i=1;i<n;i++)
{
temp = arr[i];
j = i-1;
while((temp < arr[j]) && (j>=0))
{
arr[j+1] = arr[j];
j--;
```

```
}
arr[j+1] = temp;
}
}
```

14. Write a program to sort an array using selection sort algorithm.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
int smallest(int arr[], int k, int n);
void selection_sort(int arr[], int n);
int main(int argc, char *argv[]) {
int arr[10], i, n;
printf("\n Enter the number of elements in the array: ");
scanf("%d", &n);
printf("\n Enter the elements of the array: ");
for(i=0;i<n;i++)
{
scanf("%d", &arr[i]);
}
selection_sort(arr, n);
printf("\n The sorted array is: \n");
for(i=0;i<n;i++)
printf(" %d\t", arr[i]);
}
int smallest(int arr[], int k, int n)
{
int pos = k, small=arr[k], i;
for(i=k+1;i<n;i++)
{
if(arr[i]< small)
{
small = arr[i];
pos = i;
}
}
return pos;
```

```
}
void selection_sort(int arr[],int n)
{
int k, pos, temp;
for(k=0;k<n;k++)
{
pos = smallest(arr, k, n);
temp = arr[k];
arr[k] = arr[pos];
arr[pos] = temp;
}
}
```

15. Write a program to implement shell sort algorithm.

```
#include<stdio.h>
int main()
{
int arr[10]={-1};
int i, j, n, flag = 1, gap_size, temp;
printf("\n Enter the number of elements in the array: ");
scanf("%d", &n);
printf("\n Enter %d numbers: ",n); // n was added
for(i=0;i<n;i++)
scanf("%d", &arr[i]);
gap_size = n;
while(flag == 1 || gap_size > 1)
{
flag = 0;
gap_size = (gap_size + 1) / 2;
for(i=0; i< (n - gap_size); i++)
{
if( arr[i+gap_size] < arr[i])
{
temp = arr[i+gap_size];
arr[i+gap_size] = arr[i];
arr[i] = temp;
flag = 0;
```

```
}
}
}
printf("\n The sorted array is: \n");
for(i=0;i<n;i++){
printf(" %d\t", arr[i]);
}
}
```